

ADD literal to W **ADDLW** [label] ADDLW k Syntax: $0 \le k \le 255$ Operands: Operation: $(W) + k \rightarrow W$ N,OV, C, DC, Z Status Affected: kkkk 0000 1111 kkkk Encoding: The contents of W are added to the 8-Description: bit literal 'k' and the result is placed in W. 1 Words: 1 Cycles:

Q Cycle Activity:

QI	Q2	Q3	Q4
Decode	Read	Process	Write to W
	literal 'k'	Data	<u> </u>

Example:

ADDLW 0x15

Before Instruction

W = 0x10

After Instruction

W 0x25

ADDWF	ADD W to	f		
Syntax:	[label] Al	DDWF	f,d,a	
Operands:	$0 \le f \le 259$ $d \in [0,1]$ $a \in [0,1]$	5 .		
Operation:	(W) + (f) -	→ dest		
Status Affected:	N,OV, C,	DC, Z	`	
Encoding: '	0010	01da	ffff	ffff
Description:	Add W to register "T. If "d" is 0 the is stored in W. If "d" is 1 the result stored back in register "T" (default is 0 Virtual bank will be selected. 1 the BSR will not be overridden (default).			
Words:	1		•	

Cycles:

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example:

ADDWF

REG, 0, 0

Before Instruction

0x17 W REG 0xC2

After Instruction

W 0xD9 0xC2 REG

ADD W and Carry bit to f ADDWFC [label] ADDWFC: Syntax: $0 \le f \le 255$ Operands: $d \in [0,1]$ $a \in [0,1]$ $(W) + (f) + (C) \rightarrow dest$ Operation: N,OV, C, DC, Z Status Affected: 00da ffff ffff 0010 Encoding: Add W, the Carry Flag and data memory Description: location 'f'. if 'd' is 0, the result is placed in W. If 'd' is 1, the result is placed in data memory location 'f'. If 'a' is 0 Virtual bank will be selected. If 'a' is 1 the BSR will not be overridden. Words:

Cycles: Q Cycle Activity:

O1	Q2	Q3	Q4
Decode	Read	Process	Write to
Decodo	register "	Data	destination

Example:

ADDWFC

REG, 0, 1

Before Instruction

Carry bit = 1REG = 0x02W = 0x4D

After Instruction

Carry bit = 0 REG = 0x02 W = 0x50 AND literal with W

Syntax: [label] ANDLW k

Operands: $0 \le k \le 255$ Operation: (W) AND. $k \rightarrow W$

Status Affected: N,Z

Encoding: , 0000 1011 kkkk kkkk

Description: The contents of W are AND'ed with the 8-bit literal 'k'. The result is placed in W.

Words: 1
Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4 :
Decode	Read literal	Process	Write to W
	'K	Data	

Example:

ANDLW

0x5F

Before Instruction

W = 0xA3

After Instruction

W = 0x03

AND W with f ANDWF [label] ANDWF f,d,a Syntax: $0 \le f \le 255$ Operands: $d \in [0,1]$ $a \in [0,1]$ Operation: (W) .AND. (f) \rightarrow dest N,Z Status Affected: ffff ffff 0001 01da Encoding: The contents of W are AND'ed with reg-Description: ister 'f. If 'd' is 0 the result is stored in W. If 'd' is 1 the result is stored back in register 'f' (default). If 'a' is 0 Virtual bank will be selected. If 'a' is 1 the BSR will not be overridden (default). 1 Words: 1 Cycles:

Q1	Q2	Q3	Q4
Decode	Read	Process	Write to
	register 'f'	Data	destination

REG, 0, 0

Example: ANI Before Instruction

Q Cycle Activity:

W = 0x17

ANDWF

REG = 0xC2

After Instruction

W = 0x02 REG = 0xC2

BC Branch if Carry

Syntax: [label] BC n

Operands: $-128 \le n \le 127$

Operation: if carry bit is '1'

(PC) +2+2n → PC

Status Affected: None

Encoding: 1110 0010

Description: If the Carry bit is '1', then the program

will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a

nnnn

nnnn

two-cycle instruction.

Words:

Cycles: 1(2)

Q Cycle Activity:

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1 .	Q2	Q3	Q4
Decode	Read literal	Process	No
255000	'n'	Data	operation

Example:

HERE

BC 5

Before Instruction

PC = address (HERE)

After Instruction

If Carry = 1

PC = address (HERE+12)

If Carry = 0

PC = address (HERE+2)

Bit Clear f **BCF** [label] BCF f,b,a Syntax: $0 \le f \le 255$ Operands: 0 ≤ b ≤ 7 $a \in [0,1]$ 0 → f Operation: Status Affected: None ffff ffff 1001 bbba Encoding: Bit 'b' in register 'f' is cleared. If 'a' is 0 Description: Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default). Words: Cycles: Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'P

Example:

BCF

7, 0 FLAG_REG,

Before Instruction FLAG_REG = 0xC7 After Instruction FLAG_REG = 0x47

BN	Branch if	Negativ	re		
Syntax:	[label] B	[label] BN n			
Operands:	-128 ≤ n ≤	-128 ≤ n ≤ 127			
Operation:	if negative (PC) +	e bit is '1 + 2 + 2n -			
Status Affected:	None				
Encoding:	1110	0110	nnnn	nnnn	
Description:		oranch. oranch. oranchementhe PC. Somented to, the new orange. This ins	nt number the Footetch the raddress truction is	2n' is C will next will be	
Words:	ş. 1			•	
Cuelent	1/2\				

Cycles:

1(2)

Q Cycle Activity:

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process Data	Write to PC
No operation	No operation	No operation	. No operation

If No Jump:

Decode Read literal Process	QC-1
'n' Data	No operation

Example:

HERE

Jump

Before Instruction

PC

address (HERE)

After Instruction

If Negative PC

address (Jump)

If Negative PC

0; address (HERE+2)

Branch if Not Carry BNC [label] BNC n Syntax: -128 ≤ n ≤ 127 Operands: if carry bit is '0' Operation: $(PC) + 2 + 2n \rightarrow PC$ Status Affected: None 1110 nnnn mnn 0011 Encoding: If the Carry bit is '0', then the program Description: will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction. Words: 1(2) Cycles: Q Cycle Activity: If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process	No
3000	'n'	Data	operation

Example:

HERE

ENC Jump

Before Instruction

address (HERE)

After Instruction

If Carry

address (Jump)

If Carry

address (HERE+2)

Branch if Not Negativ BNN

Syntax:

[label] BNN. n

Operands:

-128 ≤ n ≤ 127

Operation:

if negative bit is '0'

 $(PC) + 2 + 2n \rightarrow PC$

0111

Status Affected:

Description:

None

1110

Encoding:

If the Negative bit is '0', then the pro-

nnnn

nnnn

gram will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a

two-cycle instruction.

Words:

1

Cycles:

1(2)

Q Cycle Activity:

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1 .	Q2	Q3	Q4
Decode	Read literal	Process	No
00000	'n'	Data	operation

Example:

HERE

BNN Jump

Before Instruction

address (HERE) PC

After Instruction

0; If Negative

address (Jump)

If Negative PC

address (HERE+2)

INV	Branch if	Not Ov	erflow	
iyntax:	[label] B	NV n		
)perands:	-128 ≤ n ≤	127		
Operation:	if overflow bit is '0' (PC) + 2 + 2n \rightarrow PC			
Status Affected:	None			
Encoding:	1110	0101	nnnn	nnnn
Description:	If the Overflow bit is '0', then the program will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction.			
Words:	1 ·			
Cycles:	1(2)			
Q Cycle Activity: If Jump:				0.4
Q1	Q2	Q3		Q4
Decode	Read litera		cess ata	Write to PC

De	code	'n'	Data	
	No ration	No operation	No operation	No operation
If No Ju	mp:	03	O3	04

Q1	Q2	Q3	Q4
Decod	Read literal	Process Data	No operation

Example:	HERE	BNV Jump
Before Instru PC	ction =	address (HERE)
After Instruct If Overflo PC If Overflo PC	w = =	0; address (Jump) 1; address (HERE+2)

BNZ		Branch if	Not Zer	0		
Syntax	C.	[label] BNZ n				
Opera	nds:	-128 ≤ n ≤ 127				
Opera	tion:	if zero bit is '0' $(PC) + 2 + 2n \rightarrow PC$				
Status	s Affected:	None				
Encod	iing:	1110	0001	nnnn	nnnn	
Descr	iption:	If the Zero will branch		then the	program	
		The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction.				
Word	ls:	1				
Cycle	es:	1(2)				
Q Cy If Jui	cle Activity:			•		
	Q1	Q2	Q3		Q4	
	Decode	Read litera		ess ata	Write to PC	
	No operation	No operation		lo ation	No operation	

Q.	ump: I	Q2	Q3	Q4:
	Decode	Read literal	Process Data	No operation

Example:	HERE	BNZ	Jamb
Before Instruction	ction =	address	(HERE)
After Instructi	on		
If Zero PC If Zero PC	= =	0; address 1; address	(Jump) (HERE+2)

Unconditi nai Branch BRA [label] BRA n Syntax: -1024 ≤ n ≤ 1023 Operands: $(PC) + 2 + 2n \rightarrow PC$ Operation: Status Affected: None nnnn 1101 0nnn nnnn Encoding: Add the 2's complement number '2n' to Description: the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is a two-cycle instruction.

Words:

1

Cycles:

2

Q Cycle Activity:

Q1	Q2	Q3	Q4:
Decode	Read literal	Process Data	Write to PC
No operation	No operation	No operation	No operation

Example:

HERE

Jump BRA

Before Instruction

PC

address (HERE)

After Instruction

PC

address (Jump)

Bit Set f **BSF**

Syntax:

[label] BSF

Operands:

 $0 \le f \le 255$

0 ≤ b ≤ 7

 $a \in [0,1]$

Operation:

1 → f

Status Affected:

Description:

None

Encoding:

1000

bbba Bit 'b' in register 'f' is set. If 'a' is 0 Vir-

tual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will

ffff

ffff

be selected as per the BSR value.

Words:

1

Cycles:

1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

Example:

BSF

FLAG_REG, 7, 1

Before Instruction

FLAG_REG=

0x0A

After Instruction

FLAG_REG=

0x8A

BTFSC	Bit Test F	īle, Skip i	f Clear	_	В
Syntax:		[label] BTFSC f,b,a			
Operands:	$0 \le f \le 255$ $0 \le b \le 7$ $a \in [0,1]$			C	
Operation:	skip if (f<	b>) = 0			C
Status Affected:	None				S
Encoding:	1011	bbba	ffff	ffff	E
Description:	instruction If bit 'b' is fetched di cution is o cuted inst instruction selected, = 1, then	register 'f' is is skipped. O then the ruring the culliscarded, a ead, makinh. If 'a' is 0 overriding the bank walue (defau	next instruction rent instruction instruction and a NOP is gifted at two Virtual barthe BSR value be selected.	ction uction exe- is exe- o-cycle ok will be alue. If 'a'	
Words:	1				
Cycles:	1(2) Note: 3	cycles if s	kip and fo	llowed	

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

by a 2-word instruction

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
operation	operation	operation	Opei

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:	HERE	BTFSC	FLAG.	1,	0	
	FALSE	:				
	TRUE	:				
Before Instr	uction					

Retore	Instruction	
PC		

address (HERE)

After Instruction

If FLAG<1>

address (TRUE) address (FALSE)

STFSS	Bit Test Fil	, Skip if Set

[label] BTFSS f,b,a yntax:

 $0 \le f \le 255$ p rands:

0≤b<7 $a \in [0,1]$

skip if (f < b >) = 1Operation:

None Status Affected:

ffff 1010 ffff Encoding:

If bit 'b' in register 'f' is 1 then the next Description: instruction is skipped.

If bit 'b' is 1, then the next instruction fetched during the current instruction execution, is discarded and an NOP is executed instead, making this a two-cycle instruction. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per

the BSR value (default).

Words:

1(2) Cycles:

1

Note: 3 cycles if skip and followed by a 2-word instruction

Q Cycle Activity:

Q1 · Q2	2 (3)	<u> </u>
Decode	Read Proces	s Data No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q2	Q3	Q4
No operation	No operation	No operation
No	No operation	No operation
	operation	operation operation No No

FLAG, 1, 0 BTFSS Example: HERE FALSE TRUE

Before Instruction

address (HERE) PC

After Instruction

If FLAG<1>

address (FALSE)

AG<1>

address (TRUE)

Bit Toggle f BTG [label] BTG f,b,a Syntax: Operands: $0 \le f \le 255$ 0≤b<7 $a \in [0,1]$ ({}) → { Operation: Status Affected: None ffff ffff 0111 bbba Encoding: Bit 'b' in data memory location 'T is Description: inverted. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (defauit).

Words: Cycles:

Q Cycle Activity:

2	Q3	Q4
Read register "f"	Process Data	Write register "
	<u>"</u>	Read Process

Example:

BTG

PORTC, 4, 0

Before Instruction:

PORTC = 0111 0101 [0x75]

After Instruction:

0110 0101 [0x65] PORTC =

Branch if Overflow BV

Syntax:

[label] BV n

Operands:

-128 ≤ n ≤ 127

Operation:

if overflow bit is '1'

 $(PC) + 2 + 2n \rightarrow PC$

Status Affected:

None

Encoding: Description: 1110 0100 nnnn nnnn

If the Overflow bit is '1', then the program will branch.

> The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a

two-cycle instruction.

Words:

Cycles:

1 1(2)

Q Cycle Activity:

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process	No
555555	'n'	Data	operation

Example:

HERE

ΒV Jump

Before Instruction

PC

address (HERE)

After Instruction

If Overflow

address (Jump)

PC If Overflow PC

address (HERE+2)

BZ Branch if Zero

Syntax: [label] BZ n

Operands: -128 ≤ n ≤ 127

Operation: if Zero bit is '1'

(PC) + 2 + 2n → PC

Status Affected: None

Encoding: 1110 0000 nnnn nnnn

Description: If the Zero bit is '1', then the program will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a

two-cycle instruction.

Words:

1.

Cycles:

1(2)

Q Cycle Activity:

If Jump:

	Q1	Q2	Q3	Q4
[Decode	Read literal	Process Data	Write to PC
	No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process	No
000000	'n'	Data	operation

Example:

HERE

BZ Jump

Before Instruction

PC

= address (HERE)

After Instruction

If Zero =

PC = address (Jump)

If Zero = 0;

PC = address (HERE+2)

Subroutine Call

Syntax: [label] CALL k,s

Operands: $0 \le k \le 1048575$ $s \in [0,1]$ Operation: $(PC) + 4 \rightarrow TOS$, $k \rightarrow PC < 20:1 >$,
if s = 1 $(W) \rightarrow WS$, $(STATUS) \rightarrow STATUSS$, $(BSR) \rightarrow BSRS$

Status Affected: None

Encoding:

1st word (k<7:0>)
2nd word(k<19:8>)

-	110	110s k ₁₉ kkk	k ₇ kkk kkkk	kkkk ₀ kkkk ₈
---	-----	-----------------------------	----------------------------	--

Description:

Subroutine call of entire 2M byte memory range. First, return address (PC+ 4) is pushed onto the return stack. If 's' = 1, the W, STATUS and BSR registers are also pushed into their respective shadow registers, WS, STATUSS and BSRS. If 's' = 0, no update occurs (default). Then the 20-bit value 'k' is loaded into PC<20:1>. CALL is a two-

cycle instruction.

Words:

2

Cycles:

2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'<7:0>,	Push PC to stack	Read literal 'k'<19:8>, Write to PC
No operation	No operation	No operation	No operation

Example:

HERE

CALL THERE, Fast

Before Instruction

PC = Address (HERE)

After Instruction

PC = Address (THERE)
TOS = Address (HERE + 4)

WS = W BSRS= BSR

STATUSS = STATUS

Clear f CLRF [label] CLRF: f,a Syntax: 0 ≤ f ≤ 255 Operands: $a \in [0,1]$ $000h \rightarrow f$ Operation: $1 \rightarrow Z$ Z Status Affected: ffff ffff 101a 0110 Encoding: Clears the contents of the specified reg-Description: ister. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default). 1 Words: ٠1 Cycles: Q Cycle Activity: Q4 Q3 Q2 Q1 Write **Process** Read Decode

Clear Watchd g Timer CLRWDT [label] CLRWDT Syntax: Operands: None $000h \rightarrow WDT$, Operation: 000h → WDT postscaler, $1 \rightarrow \overline{10}$ 1 → PD TO, PD Status Affected: 0100 0000 0000 0000 **Encoding:** CLEWDT instruction resets the Watch-Description: dog Timer. It also resets the postscaler of the WDT. Status bits TO and PD are 1 Words: 1 Cycles: Q Cycle Activity: Q4 Q3 Q2 Q1 No **Process** Decode No operation operation Data

register T Data register T

Example: CLRF FLAG_REG, 1

Before Instruction

 $FLAG_REG = 0x5A$

After Instruction

 $FLAG_REG = 0x00$

Example:

CLRWDT

COMF	Complem	ent f		
Syntax:	[label] (COMF	f,d,a	
Operands:	$0 \le f \le 25$ $d \in [0,1]$ $a \in [0,1]$	5		
Operation:	$(\vec{f}) \rightarrow d$	est		
Status Affected:	N,Z			
Encoding:	0001	11da	ffff	ffff
Description:	The contents of register 'f' are complemented. If 'd' is 0 the result is stored in W. If 'd' is 1 the result is stored back in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).			
Words:	1			
Cycles:	1			
Q Cycl Activity:				<u> </u>
Q1	Q2	Q3		24
Decode	Read	Pro	cess	Write to

Example: COMF

Before Instruction

REG = 0x13

After Instruction

REG = 0x13

W

= 0x13 = 0xEC

register 'f

Data

REG, 0, 0

Compare f with W, skip if f = W **CPFSEQ** [label] CPFSEQ f,a Syntax: $0 \le f \le 255$ Operands: $a \in [0,1]$ Operation: (f) - (W),skip if (f) = (W)(unsigned comparison) Status Affected: None ffff 001a ffff 0110 **Encoding:** Compares the contents of data memory Description: location 'f' to the contents of W by performing an unsigned subtraction. If T = W then the fetched instruction is discarded and an NOP is executed instead making this a two-cycle instruction. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed

by a 2-word instruction

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register f	Process Data	No operation

per the BSR value (default).

If skip:

destination

Q2	Q3	Q4
No operation	No operation	No operation
		No No

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

HERE CPFSEQ REG, 0

NEQUAL

EQUAL

Before Instruction

PC Address = HERE W = ? REG = ?

After Instruction

If REG = W;

PC = Address (EQUAL)

If REG # W;

PC = Address (NEQUAL)

CPFSGT	Compare f with W, skip if f > W			
Syntax:	[label] C	PFSGT	f,a	
Operands:	$0 \le f \le 25$ $a \in [0,1]$	5		
Operation:	(f) - (W), skip if (f) > (unsigned		on)	
Status Affected:	None			
Encoding:	0110	010a	ffff	ffff
Description:	instruction selected,	to the cor g an unsignents of The Withen carded aread making and the in- termination of the in- termination of the bare of the	ntents of the prediction of the fetche and NOP of this a to the the the thing of the	ne W by action. or than the d instruc- is exe- wo-cycle ank will be
Words:	1			
Cycles:			skip and d instruct	

Q Cycle Activity:

•	Q1	Q2	Q3	Q4
	Decode	Read	Process	No operation
		register "	Data	operation

If skip:

Q1	Q2	Q3	Q4
No	No	No	No
operation	operation	operation	operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

HERE

CPFSGT REG, 0

NGREATER

GREATER

Before Instruction

Address (HERE) PC

W

After Instruction

W: If REG

Address (GREATER)

W; If REG

Address (NGREATER)

CPFSLT	Compare	f with W	, skip if	t < W
Syntax:	[label]	CPFSLT	f,a	
Operands:	$0 \le f \le 25$ $a \in [0,1]$	5		
Operation:	(f) – (W), skip if (f) < (unsigned		on)	
Status Affected:	None			
Encoding:	0110	000a	ffff	ffff
Description:	Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction.			
	tion is dis cuted inst instruction	of W, then carded an tead makin n. If 'a' is	are less the the fetcher an NOP and this a the O Virtual bette BSR v	ed instruction is exe- wo-cycle ank will b

Words:

Cycles:

Note: 3 cycles if skip and followed

by a 2-word instruction

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register "f	Process Data	No operation

overridden (default).

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

CPFSLT REG, 1 HERE Example: NLESS

LESS

Before Instruction

Address (HERE) PC W

After Instruction

W; If REG

Address (LESS) PC

If REG ≥

Address (NLESS) PC

DAW	Decimal Adjust W Register					
Syntax:	[label] DAW					
Operands:	None					
Op ration:	If [W<3:0> >9] or [DC = 1] then (W<3:0>) + 6 → W<3:0>; else					
	(W<3:0>) → W<3:0>;					
	If [W<7:4> (W<7:4 else (W<7:4	>9] or [C I>) + 6 → I>) → W<				
Status Affected:	C					
Encoding:	0000	0000	0000	0111		
Description:	DAW adjuresuiting from variables and produresuit.	rom the ea (each in p	artier addit acked BC	ion of two D format)		
Words:	1					
Cycles:	1					

Q Cycle Activit	y:			
Q1	Q2	Q3	Q4	_
Decode	Read register W	Process Data	Write W	

	QI	QZ_	GC		
	Decode	Read register W	Process Data	Write W	
Exa	mple1:	DAW			

Before Instruction

0xA5 W 0 C DC

After Instruction

0x05

Example 2:

Before Instruction

W = 0xCE C

After Instruction

W C DC 0x34

DECF		Decrem r	nt f		
Syntax:	_	[label] [ECF f,	i,a	
Operands:		$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$	5		
Operation:		$(f)-1\rightarrow 0$	dest		
Status Affe	ected:	C,DC,N,C	V,Z		
Encoding:		0000	01da	ffff	ffff
Description	n:	Decrement result is stored by 'a' is 0 Viril overriding then the by the BSR v	ored in W ack in req wal bank the BSR ank will b	/. If 'd' is 1 gister 'f' (d will be se value. If e selecte	the result efault). If lected, 'a' = 1,
Words:		1			
Cycles:		11			
Q Cycle A	Activity:				
Q1		Q2	Q3	C	14
De	code	Read	Proc	ess	Write to

Decode Re regis	ad Process ter 'f Data	Write to destination

CNT, 1, 0 DECF Example: Before Instruction 0x01 0 After Instruction

DECFSZ	Decrement f,	skip if 0			
Syntax:	[label] DEC	FSZ f,d,a			
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$				
Operation:	(f) $-1 \rightarrow des$ skip if result :				
Status Affected:	None				
Encoding:	0010 11	da ffff	ffff		
Description:	The contents of register 'f are decremented. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed back in register 'f' (default). If the result is 0, the next instruction, which is already fetched, is discarded, and an NOP is executed instead making it a two-cycle instruction. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).				
Words:	1				
Cycles:	4/0)				
Q Cycle Activity:					
Q1	Q2	Q3	Q4		
Decode	Read register 'f'	Process Data	Write to destination		
If skip:					
Q1	Q2	Q3	Q4		
No	No operation	No operation	No operation		
operation If skip and follov					
Q1	Q2	Q3	Q4		
No	No	No	No		
operation	operation	operation	operation		
No	No	No	No operation		
operation	operation	operation	operation		
Example:	HERE	DECFSZ GOTO	CNT, 1, 1 LOOP		
	CONTINUE	:			
Before Ins		ss (HERE)			

After Instruction

CNT =
If CNT =
PC =
If CNT #
PC =

CNT - 1 0: Address (CONTINUE) 0; Address (HERE+2)

		•				
DCFSN		Decremer	_)
Syntax:		[label] D	CFS	NZ	f,d,a	
Operan	.	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$				
Operati	on:	(f) $-1 \rightarrow 0$ skip if res			•	
Status	Affected:	None				
Encodi	ng:	0100	11	.da	ffff	ffff
Descrip	otion:	The contents of register 'f are decremented. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed back in register 'f' (default). If the result is not 0, the next instruction, which is already fetched, is discarded, and an NOP is executed instead making it a two-cycle instruction. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected				
		as per the	BS	R va	lue(defau	ılt).
Words	s:	1				
Cycle	s:	1(2) Note: 3 by	cycl a 2	es if	skip and	d followed ction
Q Cyc	cle Activity:					
	Q1	Q2	. (Q3_		Q4
	Decode	Read register f			cess - ata	Write to destination
lf skip	D:					
	Q1	Q2		Q3		Q4
	No	No			No	No operation
operation operation operation operation If skip and followed by 2-word instruction:					орстанон	
			uction.	Q4		
ı	Q1 No	No.		40	No	No
	operation	operatio	n	орі	eration	operation
	No	No			No	No .

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:	HERE ZERO NZERO	: :	PSNZ	TEMP, 1,	0
Before Instru	ction	=	?		
After Instruction TEMP If TEMP PC If TEMP PC PC			0;	P-1, ess (ZERO) ess (NZERO)	١.

Unconditional Branch GOTO [label] GOTO k Syntax: 0 ≤ k ≤ 1048575 Operands: k → PC<20:1> Operation: None Status Affected: Encoding: $kkkk_0$ 1111 k7kkk 1st word (k<7:0>) 1110 kkkk₈ k₁₉kkk kkkk 2nd word(k<19:8>) 1111 GOTO allows an unconditional branch Description: anywhere within entire 2M byte memory

cycle instruction.

2

Cycles:

Words:

2

Q Cycle Activity:

	Q1	Q2	Q3	Q4
	Decode	Read literal 'k'<7:0>,	No operation	Read literal 'k'<19:8>, Write to PC
Ì	No operation	No operation	No operation	No operation

range. The 20-bit value 'k' is loaded into PC<20:1>. GOTO is always a two-

Example:

GOTO THERE

After Instruction

PC = Address (THERE)

HALT	Halt Prod	essor		
Syntax:	[label]	HALT		
Operands:	None		•	
Operation:	Processo HALT ins		kecution a	after
Status Affected:	None			
Encoding:	0000	0000	0000	0001
Description:	processor HALT pin bring the	of the hain rexecution or resetting device our n is not re-	t instruction. Toggiing ng (MCLR t of hait. H	n will hal the = 0) will ALT
Words:	1,			
Cycles:	1	•		

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No	No	HALT
	operation	operation	

INCF	Increment f				
Syntax:	[label] INCF f,d,a				
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$				
Operation:	$(f) + 1 \rightarrow dest$				
Status Affected:	C,DC,N,OV,Z				
Encoding:	0010 10da ffff ffff				
Description:	The contents of register 'f' are incremented. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed back in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).				
Words:	1				
Cycles:	1				

QC	cle Activity:	:		
	Q1	Q2	Q3	Q4
	Decode	Read register 'f'	Process Data	Write to destination

CNT, 1, 0

Example:	INCF

Before Instruction CNT Z C DC 0xFF 0?? =

After Instruction

CNT Z C DC 0x00 =

INCFSZ	Increment f, skip if 0					
Syntax:	[label]	INCFSZ	f,d,a			
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$	5				
Operation:	(f) + 1 \rightarrow skip if res					
Status Affected:	None					
Encoding:	0011	11da	ffff	ffff		
Description:	The contents of register T are incremented. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed back in register T. (default) If the result is 0, the next instruction, which is already fetched, is discarded, and an NOP is executed instead making it a two-cycle instruction. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).					
Words:	1					
Cycles:	1(2) Note: 3	cycles if	skip and	followed		

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read	Process	Write to
	register 'f'	Data	destination

by a 2-word instruction

If skip:

No No No No No operation operation	Q1	Q2	Q3	Q4
	No operation	No operation	1	, ,,,

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

HERE NZERO ZERO

INCFSZ

CNT, 1, 0

Before Instruction

Address (HERE) PC

After Instruction

CNT + 1

0; Address (ZERO)

0; Address (NZERO)

Increment f, skip if not 0 INFSNZ INFSNZ f,d,a [label] Syntax: 0 ≤ f ≤ 255 Operands: $d \in [0,1]$ $a \in [0,1]$ (f) + 1 \rightarrow dest, Operation: skip if result ≠ 0 Status Affected: None ffff 10da ffff 0100 **Encoding:** The contents of register 'f' are incre-Description:

mented. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed back in register 'f' (default).

If the result is not 0, the next instruction, which is already fetched, is discarded, and an NOP is executed instead making it a two-cycle instruction. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value

(default).

Words:

1

Cycles:

1(2)

Note: 3 cycles if skip and followed by a 2-word instruction

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

If skip:

Q1		Q2	Q3	Q4
	No	No	No	No
on	eration	operation	operation	operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

HERE

INFSNZ REG, 1, 0

ZERO

NZERO

B fore Instruction

Address (HERE) PC

After Instruction

REG + 1 REG

0; If REG

Address (NZERO)

Address (ZERO)

Inclusive OR literal with W IORLW

[label] IORLW k. Syntax:

 $0 \le k \le 255$ Operands:

Operation: (W) .OR. $k \rightarrow W$

N,Z Status Affected:

kkkk 0000 1001 kkkk **Encoding:**

The contents of W are OR'ed with the Description: eight bit literal 'k'. The result is placed in

1 Words: 1

Cycles: Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read	Process	Write to W
	literal 'K'	Data	

Example:

IORLW

0x35

Before Instruction

W 0x9A

After Instruction

0xBF

Inclusive OR W with f IORWF [label] IORWF f,d,a Syntax: $0 \le f \le 255$ Operands: $d \in [0,1]$ $a \in [0,1]$ (W) .OR. (f) \rightarrow dest Operation: Status Affected: N.Z ffff ffff 0001 00đa Encoding: Inclusive OR W with register 'f'. If 'd' is 0 Description: the result is placed in W. If 'd' is 1 the result is placed back in register T (defauit). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default). 1 Words: 1 Cycles: Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: IORWF RESULT, 0, 1

Before Instruction RESULT = 0x13

W = 0x91

After Instruction

RESULT = 0x13

W = 0x93

MOVF M v f

Syntax: [label] MOVF f,d,a.

Operands: $0 \le f \le 255$

 $d \in [0,1]$ $a \in [0,1]$

Operation: $f \rightarrow dest$

Status Affected: N,Z

Encoding: 0101 00da fffff ffff

Description: The contents of register Y is moved to a

destination dependent upon the status of 'd'. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed back in register 'f' (default). Location 'f' can be anywhere in the 256 byte bank. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank

will be selected as per the BSR value

(default).

Words:

1

Cycles:

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read	Process	Write W
	register 4	Data	

Example: MOVF REG, 0, 0

Before Instruction

REG = 0x22W = 0xFF

After Instruction

REG = 0x22

W = 0x22

MOVFF

Move f to f

Syntax:

[label] MOVFF fs,fd

Operands:

 $0 \le f_s \le 4095$

 $0 \le f_d \le 4095$

Operation:

 $(f_s) \rightarrow f_d$

Status Affected:

None

Encoding: 1st word (source) 2nd word (destin.)

	LLLL	a
\ 1111 ffff	ffff	ffffa
1100 ffff	ffff	ffffg

Description:

The contents of source register 'fs' are moved to destination register 'fd'. Location of source 'fs' can be anywhere in the 4096 byte data space (000h to FFFh), and location of destination 'fd' can also be anywhere from 000h to FFFh.

Either source or destination can be W (a useful special situation).

MOVFF is particularly useful for transferring a data memory location to a peripheral register (such as the transmit buffer or an I/O port).

The MOVEP instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register

Words:

2

Cycles:

2 (3)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f' (src)	Process Data	No operation
Decode	No operation No dummy read	No operation	Write register 'f' (dest)

Example:

.REG1, REG2

Before Instruction

REG1 REG2 0x33 0x11

After Instruction

REG1 REG2 MOVLB

Move literal to low nibble in BSR

Syntax:

[label]: MOVLB k.

Operands:

 $0 \le k \le 255$

Operation:

k → BSR

Status Affected:

None

Encoding:

0000 0001 kkkk

Description:

The 8-bit literal 'k' is loaded into the

Bank Select Register (BSR).

Words:

Cycles:

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal	Process Data	Write literal 'k' to BSR

0x02

Example:

5 MOVLB

Before Instruction

BSR register =

After Instruction

0x05 BSR register =

Move literal t FSR LFSR [label] LFSR f,k. Syntax: 0≤f≤2 Operands: 0 ≤ k ≤ 4095 Operation: k → FSRf Status Affected: None k₁₁kkk 1110 OOff 1110 Encoding: kkkk 0000 k7kkk 1111 The 12-bit literal 'k' is loaded into the Description: file select register pointed to by 'f' 2 Words: 2 Cycles: Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k' MSB	Process Data	Write literal 'k' MSB to FSRfH
Decode	Read literal	Process Data	Write literal

Example:

LFSR 2, 0x3AB

After Instruction

FSR2H = 0x03FSR2L = 0xAB

Move literal to W MOVLW [label], MOVLW k Syntax: 0 ≤ k ≤ 255 Operands: Operation: $k \rightarrow W$ None Status Affected: kkkk kkkk 0000 1110 Encoding: The eight bit literal 'k' is loaded into W. Description: Words: 1 Cycles:

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read	Process	Write to W
	literal 'k'	Data	<u> </u>

Example:

MOVLW

0x5A

After Instruction

W = 0x5A

M ve W to f MOVWF MOVWF Syntax: [label] $0 \le f \le 255$ Operands: $a \in [0,1]$ Operation: $(W) \rightarrow f$ Status Affected: None ffff 0110 111a Encoding: Move data from W to register f. Loca-Description: tion 'f' can be anywhere in the 256 byte bank. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as

Words:

Cycles:

1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

per the BSR value (default).

Example:

MOVWF

REG, 0

Before Instruction

W 0x4F 0xFF REG

After Instruction

0x4F W REG 0x4F MULLW Multiply Literal with W

[label] MULLW Syntax:

Operands: $0 \le k \le 255$

Operation: (W) $x k \rightarrow PRODH:PRODL$

Status Affected: None

Encoding:

kkkk 0000 1101 kkkk

An unsigned multiplication is carried Description: out between the contents of W and the 8-bit literal 'k'. The 16-bit result is placed in PRODH:PRODL register pair. PRODH contains the high byte.

W is unchanged.

None of the status flags are affected. Note that neither overflow nor carry

is possible in this operation. A zero result is possible but not detected.

Words:

1

Cycles:

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write registers PRODH: PRODL

Example:

0xC4

Before Instruction

0xE2 W PRODH PRODL

After Instruction

0xE2 0xAD **PRODH** 80x0 PRODL

Multiply W with f MULWF [label] MULWF Syntax: $0 \le f \le 255$ Operands: $a \in [0,1]$ (W) \times (f) \rightarrow PRODH:PRODL Operation: None Status Affected: ffff 0000 001a Encoding: An unsigned multiplication is carried Description: out between the contents of W and the register file location "f. The 16-bit result is stored in the PRODH:PRODL register pair. PRODH contains the high byte. Both W and 'f' are unchanged. None of the status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default). 1 Words:

Cycles: Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write registers PRODH: PRODL

Example:

MULWF REG, 1

Before Instruction

W = 0xC4 REG = 0xB5 PRODH = ? PRODL = ?

1

After Instruction

W = 0xC4
REG = 0xB5
PRODH = 0x8A
PRODL = 0x94

NEGF		Negate	f		
Syntax	C	[label]	NEGF	f,a	
Opera	nds:	0≤f≤2 a∈[0,1			
Opera	ition:	(f)+1	$\rightarrow f$		
Status	s Affected:	N,OV, C	C, DC, Z		
Encod	ding:	0110	110a	ffff	ffff
Desc	ription:	plement memory bank wi BSR va	The result location for the select	t is placed '. If 'a' is (ed, overrions 1, then the	ding the ne bank will
Word	ls:	1			
Cycle	es:	. 1			
Q C)	cle Activity:				
	Q1	Q2	Q3		24
	Decode	Read register		cess ata	Write register 'f

Example:

NEGF

REG, 1

Before Instruction

REG = 0011 1010 [0x3A]

After Instruction

REG = $1100 \ 0110 \ [0xC6]$

N Operation NOP NOP [label]. Syntax: None Operands: No operation Operation: None Status Affected: 0000 0000 0000 0000 **Encoding:** 200000 200000 200000 1111 No operation. Descripti n: 1 Words: Cycles: Q Cycle Activity: Q4 **Q2** Q3 Q1 No No No Decode operation operation operation

Example:

None.

P p T p of Return Stack POP [label] POP Syntax: None Operands: (TOS) → bit bucket Operation: None Status Affected: 0000 0000 0110 0000 **Encoding:** The TOS value is pulled off the return Description: stack and is discarded. The TOS value then becomes the previous value that was pushed onto the return stack. This instruction is provided to enable the user to properly manage the return stack to incorporate a software stack. Words: Cycles: · Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No	POP TOS	No
	operation	value	operation

Example:

POP

GOTO

NEW

Before Instruction

0031A2h 014332h

TOS Stack (1 level down)

After Instruction

TOS PC

014332h

NEW

Push Top of Return Stack **PUSH** [label] PUSH Syntax: None Operands: (PC+2) → TOS Operation: None Status Affected: 0101 0000 0000 0000 Encoding: The PC+2 is pushed onto the top of the Description: return stack. The previous TOS value is pushed down on the stack.

This instruction allows to implement a software stack by modifying TOS, and then push it onto the return stack.

Words: Cycles:

1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	PUSH PC+2 onto return stack	No operation	No operation

Example:

PUSH

Before Instruction

TOS = 00345Ah PC = 000124h

After Instruction

PC = 000126h TOS = 000126h Stack (1 level down) = 00345Ah RCALL Branch Subroutine

Syntax: [label] RCALL n

Operands: -1024 ≤ n ≤ 1023

Operation: $(PC) + 2 \rightarrow TOS$,

 $(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding: 1101 1nnn nnnn nnnn

Description:

Subroutine call with a jump upto 1K from the current location. First, return address (PC+2) is pushed onto the stack. Then, add the 2's complement number '2n' to the PC. Since the PC

number '2n' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is a two-

cycle instruction.

Words:

1

Cycles:

2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'n' Push PC to stack	Process Data	Write to PC
No operation	No operation	No operation	No operation

Example:

HERE

RCALL Jump

Before instruction

PC = Address (HERE)

After Instruction

PC = Address (Jump) TOS = Address (HERE+2)

Reset RESET [label] RESET Syntax: Operands: None Reset all registers and flags that Operation: are affected by a MCLR reset. All Status Affected: 0000 1111 1111 0000 **Encoding:** This instruction provides a way to exe-Description: cute a MCLR reset in software. Words: Cycles:

Q4 Q3 Q1 Q2 No No Start Decode operation reset operation

Example:

Q Cycle Activity:

RESET

After Instruction

Reset Value Registers = Reset Value Flags*

Return from Interrupt RETFIE [label] RETFIE s Syntax: $s \in [0,1]$ Operands: (TOS) \rightarrow PC, Operation: 1 → GIE/GIEH or PEIE/GIEL, if s = 1 $(WS) \rightarrow W$ (STATUSS) → STATUS, $(BSRS) \rightarrow BSR,$ PCLATU, PCLATH are unchanged. GIE/GIEH, PEIE/GIEL, STATUS reg. Status Affected: 0001 0000 0000 Encoding: Return from Interrupt. Stack is popped Description: and Top of Stack (TOS) is loaded into the PC. Interrupts are enabled by setting the either the high or low priority global interrupt enable bit. If 's' = 1, the contents of the shadow registers WS, STATUSS and BSRS are loaded into

Words: 1 2

Q Cycle Activity:

Cycles:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	pop PC from stack Set GIEH or GIEL
No operation	No operation	No operation	No operation

their corresponding registers, W, STA-

TUS and BSR. If 's' = 0, no update of

these registers occurs (default).

Example:

RETFIE Fast

After Interrupt

TOS PC WS **BSRS BSR STATUSS STATUS** GIE/GIEH, PEIE/GIEL

Rotate Left f through Carry RLCF f,d,a [label] RLCF Syntax: $0 \le f \le 255$ Operands: $d \in [0,1]$ $a \in [0,1]$ $(f<n>) \rightarrow dest<n+1>,$ Operation: $(f<7>) \rightarrow C$, (C) \rightarrow dest<0> C,N,Z Status Affected: ffff ffff 0011 01da Encoding: The contents of register "f" are rotated Description: one bit to the left through the Carry Flag. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is stored back in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

C

Words:

1

Cycles:

1

Q Cycle Activity:

Decode Read Process Write to register 1 Data destination	Ω1	Q2	Q3	Q4
	Decode	Read register 'f'		

Example:

RLCF

REG, 0, 0

register f

Before Instruction

REG C 1110 0110

After Instruction

1110 0110 REG 1100 1100 W

С

Rotate Left f (no carry) RLNCF [label] RLNCF Syntax:

Operands:

 $0 \le f \le 255$ $d \in [0,1]$

 $a \in [0,1]$

Operation:

 $(f < n >) \rightarrow dest < n+1 >$

(f<7>) → dest<0>

Status Affected:

N,Z

Encoding:

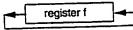
Description:

0100 01da ffff The contents of register 'P are rotated

ffff

one bit to the left. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is stored back in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value

(default).



Words:

Cycles:

1

1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example:

RLNCF

REG, 1, 0

Before Instruction

1010 1011 REG

After Instruction

0101 0111 REG

Rotate Right f through Carry RRCF [label] RRCF f,d,a Syntax: $0 \le f \le 255$ Operands: $d \in [0,1]$ $a \in [0,1]$ $(f<n>) \rightarrow dest<n-1>,$ Operation: $(f<0>)\rightarrow C,$ (C) → dest<7> C,N,Z Status Affect d: ffff ffff 0011 00da Encoding: The contents of register 'f' are rotated Description: one bit to the right through the Carry Flag. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed back in register 'f' (defauit). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be · selected as per the BSR value (default). register f 1 Words: 1 Cycles: Q Cycle Activity: Q4 Q3 Q2 Q1 Write to **Process** Read Decode destination Data register 4 REG, 0, 0 RRCF Example:

1110 0110

1110 0110

0111 0011

0

Before Instruction

After Instruction

REG

W С

REG

```
Rotate Right f (no carry)
RRNCF
                     [label] RRNCF f,d,a.
Syntax:
                     0 \le f \le 255
Operands:
                     d \in [0,1]
                      a \in [0,1]
                      (f < n >) \rightarrow dest < n-1 >,
Operation:
                      (f<0>) \rightarrow dest<7>
                      N,Z
Status Affected:
                                                       ffff
                                   00da
                                             ffff
                        0100
 Encoding:
                      The contents of register 'P are rotated
 Description:
                      one bit to the right. If 'd' is 0 the result is
                      placed in W. If 'd' is 1 the result is
                       placed back in register 'f' (default). If 'a'
                       is 0 Virtual bank will be selected, over-
                       riding the BSR value. If 'a' is 1, then the
                       bank will be selected as per the BSR
                       value (defauit).
                                          register f
                       1
  Words:
                        1
  Cycles:
  Q Cycle Activity:
                                                   Q4
                                     Q3
                      Q2
        Q1
                                                     Write to
                                       Process
          Decode
                          Read
                                                    destination
                                        Data
                       register "f.
                                   REG, 1, 0
                        RRNCF
   Example 1:
        Before Instruction
                            1101 0111
             REG
         After Instruction
                             1110 1011
             REG
                                    REG, 0,
                         RRNCF
```

1101 0111

1110 1011

1101 0111

Example 2:

Before Instruction

W

REG After Instruction

W

REG

Set f SETF [label] SETF f,a Syntax: $0 \le f \le 255$ Operands: $a \in [0,1]$ $FFh \mapsto f'$ Operation: None Status Affected: ffff ffff 0110 100a Encoding: The contents of the specified register Description: are set to FFh. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (defauit). 1 Words: 1 Cycles: Q Cycle Activity: Q4 **Q2** Q3 Q1 Write **Process** Read Decode register 'f' register " Data

SETF

Before Instruction REG

After Instruction

REG

Example:

REG, 1

0x5A

0xFF

Enter SLEEP mode SLEEP [label] SLEEP Syntax: None Operands: 00h → WDT, Operation: 0 → WDT postscaler, 1 → 10, $0 \rightarrow \overline{PD}$ TO, PD Status Affected: 0011 0000 0000 0000 Encoding: The power-down status bit (PD) is Description: cleared. The time-out status bit (TO) is set. Watchdog Timer and its postscaler are cleared. The processor is put into SLEEP mode with the oscillator stopped. Words: Cycles: Q Cycle Activity: Q4 Q3 Q2 Q1 Go to **Process** No Decode sleep Data operation

Example:

SLEEP

Before Instruction

After Instruction

TO = PD =

† If WDT causes wake-up, this bit is cleared

Subtract f from W with borrow SUBFWB [label] SUBFWB f,d,a. Syntax: 0≤f≤255 Operands: $d \in [0,1]$ $a \in [0,1]$ $(W)-(f)-(\overline{C})\to dest$ Operation: N,OV, C, DC, Z Status Affected: ffff 01da 010i Encoding: Subtract register 'f' and carry flag (bor-Description: row) from W (2's complement method). If 'd' is 0 the result is stored in W. If 'd' is 1 the result is stored in register 'f' (default) . If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default). 1 Words: 1 Cycles:

Q4

Write to

destination

Q3

Process

Data

Q Cycle Activity:

Decode

Q1

Q2

Read

SUBFWB	
Example 1:	SUBFWB REG, 1, 0
Before Instru	uction
REG	= 3
W	= 2
С	= 1
After Instruc	
REG	= FF
w	= 2 = 0
C Z N	- 0
Ñ	= 1 ; result is negative
	2 2
Example 2:	SUBFWB REG, 0, 0
Before Inst	ruction ·
REG	= 2
W	= 5
C	= 1
After Instru	
REG	= 2
W	= 3
C Z N	= 1 = 0
Ñ	= 0 ; result is positive
Example 3:	SUBFWB REG, 1, 0
Before Ins	struction ·
REG	= 1 ,
W	= 2 .
С	= 0
After Instr	_
REG	= 0
W	= 2
C Z N	= 1 = 1 ; result is zero
N	= 0

Subtract W from literal **SUBLW** [label] SUBLW k. Syntax: $0 \le k \le 255$ Operands: Operation: $k - (W) \rightarrow W$ N,OV, C, DC, Z Status Affected: 1000 kkkk 0000 Encoding: W is subtracted from the eight bit lit-Description: eral 'k'. The result is placed in W. Words: Cycles: Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to W
<u></u>			

0x02

Before Instruction

W = 1 C = 3

After Instruction

W = 1
C = 1 ; result is positive
Z = 0
N = 0

SUBLW

Example 2:

Example 1:

SUBLW 0x02

Before Instruction

W = 2 C = ?

After Instruction

W = 0 C = 1 ; result is zero Z = 1 N = 0

Example 3:

SUBLW 0x02

Before Instruction

W = 3

After Instruction

SUBWF	Subtract W from f				
Syntax:	[label] SUBWF	[label] SUBWF f,d,a			
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]				
Operation: Status Affected:	(f) – (W) → dest N.OV, C, DC, Z				
Encoding:	0101 11da	ffff	ffff		
Description:	Subtract W from register 'f' (2's complement method). If 'd' is 0 the result is stored in W. If 'd' is 1 the result is stored back in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).				
Mordo:	54 C 4 1				

Words:

Cycles:

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example 1: ST	UEWF REG, 1, 0
Before Instruction	1
REG =	3
W = C =	2 ?
After Instruction	
REG = W =	1 2
	1 ; result is positive
C = Z = N =	0
Example 2:	SUBWF REG, 0, 0
Before Instruction	
REG =	2
W = C. =	?
After Instruction	
REG = W =	2 0
	; result is zero
C = Z = N =	1. 0
Example 3:	SUBWF REG, 1, 0
Before Instruction	on
REG =	1
W = C =	2 ?
After Instruction	1
REG =	
W =	0 ; result is negative
C = Z = N =	0 1
SUBWFB	Subtract W from f with Borrow
Syntax:	[label] SUBWFB f,d,a
Op rands:	0 ≤ f ≤ 255
	$d \in [0,1]$ $a \in [0,1]$
Operation:	$a \in [0,1]$ $(f) - (W) - (\overline{C}) \rightarrow \text{dest}$
Status Affected:	N,OV, C, DC, Z
	0101 10da ffff ffff
Encoding:	Subtract W and the carry flag (borrow)
Description:	from register 'f' (2's complement
	method). If 'd' is 0 the result is stored in W. If 'd' is 1 the result is stored back
	in register 'f' (default). If 'a' is 0 Virtual
	bank will be selected, overriding the
	BSR value. If 'a' is 1, then the bank will be selected as per the BSR value
	(d fault).
Words:	1

Cycles:

Q Cycle Activity:

Q1	Q2	Q3	Q4	
Decode	Read register 'f'	Process Data	Write to destination	

SUBWFB			
Example 1:	SU	BWFB	REG, 1, 0
Before Instru	iction		
REG	=	0x19	(0001 1001)
W	=	0x0D	(0000 1101)
С	=	1	
After Instruc	tion		
REG		0x0C	(0000 1011)
W	=	0x0D	(0000 1101)
C	=	1	
C Z N	=	0	; result is positive
Example2:	= ~		REG, 0, 0
Examples.	3	OBWED	
Before Instr			
REG			(0001 1011)
W	=	0x1A	(0001 1010)
С	=	0	
After Instru	ction		
REG	=	0x1B	(0001 1011)
W	=	0x00	
C	=		and to more
C Z N	=	1	; result is zero
Example3:		_	REG, 1, 0
Before Inst			
REG			
W	=		(0000 1101)
, C	=	1	
After Instru	ıction		
REG	=		
W	=		(0000 1101)
Ç	=		
C Z N	=	0 1	; result is negative
,,,	_	•	. •

SWAP	F	Swap f			
Syntax		[label] S	WAPF	f,d,a	
Operar	nds:	$0 \le f \le 25$! $d \in [0,1]$ $a \in [0,1]$			
Operat	tion:	(f<3:0>) - (f<7:4>) -			
Status	Affected:	None			
Encod	ling:	0011	10da	ffff	ffff
Descr	iption:	ff are excluded in placed in Virtual ba	nanged. If W. If 'd' is register 'f nk will be value. If 'i	i'd' is 0 th 1 the resi' (defauit) selected a' is 1, the	. If 'a' is 0 , overriding en the bank
Word	ls:	1			
Cycle	es:	1			
Q Cy	cle Activity:				
	Q1	Q2	Q3		Q4
ļ	Decode	Read		cess	Write to

Example:

SWAPF REG, 1, 0

Data

destination

Before Instruction

REG = 0x53

register 4

After Instruction

REG = 0x35

Table Read TBLRD TBLRD (*; *+; *-; +*) [label] Syntax: None Operands: if TBLRD *, Operation: (Prog Mem (TBLPTR)) → TABLAT; TBLPTR - No Change; if TBLRD *+, (Prog Mem (TBLPTR)) → TABLAT; (TBLPTR) +1 → TBLPTR; if TBLRD *-, (Prog Mem (TBLPTR)) → TABLAT; (TBLPTR) -1 \rightarrow TBLPTR; if TBLRD +*, (TBLPTR) +1 → TBLPTR; (Prog Mem (TBLPTR)) → TABLAT; Status Affected: None 0000 10nn 0000 0000 **Encoding:** nn=0 * =1 *

There are four options with a TBLRD

Description:

There are four options with a TBLRD instruction to determine what happens to the 21-bit Table Pointer (TBLPTR): no change, post-increment, post-decrement and pre-increment. The current option is determined and the TBLPTR is modified appropriately, and the contents of the program memory location pointed to by the TBLPTR are loaded into the 8-bit Table Latch (TABLAT). The LSb of the TBLPTR selects which byte of the program memory location will be read. If LSb = 1, the high byte will be loaded into the TABLAT. If LSb = 0, the low byte will be loaded into the

TABLAT.

Words:

1

Cycles:

2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation
No operation	No operation (Table Pointer on Address bus)	No operation	No operation (OE goes low) TABLAT updated

Table Read TBLRD Example1: TBLRD Before Instruction 0x55 TABLAT 0x00A356 TBLPTR MEMORY(0x00A356) 0x34 After Instruction 0x34 TABLAT TBLPTR 0x00A357 TBLRD Example2: **Before Instruction** 0xAA 0x01A357 TABLAT TBLPTR MEMORY(0x01A357) MEMORY(0x01A358) 0x12 0x34 After Instruction 0x34 TABLAT TBLPTR 0x01A358

BLWT	Table Write
Syntax:	[label] TBLWT (*; *+; *-; +*)
Operands:	None
Operation:	if TBLWT *, (TABLAT) → Prog Mem(TBLPTR); TBLPTR - No Change; if TBLWT *+, (TABLAT) → Prog Mem(TBLPTR); (TBLPTR) +1 → TBLPTR; if TBLWT *-, (TABLAT) → Prog Mem(TBLPTR); (TBLPTR) -1 → TBLPTR; if TBLWT +*, (TBLPTR) +1 → TBLPTR; (TBLPTR) +1 → TBLPTR;
Status Affected:	None

0000

Description:

Encoding:

There are four options with a TBLWT instruction. These options determine what happens to the Table Pointer (TBLPTR): no change, post-increment, post-decrement and pre-increment. The current option is determined and the TBLPTR is modified appropriately.

0000

0000

11nn

nn=0 * =1 *+ =2 *-=3 +*

The contents of Table Latch (TABLAT) are written to the program memory location pointed to by TBLPTR.

If TBLPTR points to an external program memory location, then the instruction executes in two cycles.

Since the TABLAT is only one byte wide, a multiple of two TBLWT instructions must be executed to program internal memory locations. For example, if the device is defined to program one word at time, an internal memory location is programed in the following manner:

- 1) Set TBLPTR to an even byte
- 2) Write low byte to TABLAT
- 3) Execute TBLWT *+ (2-cycle)
- 4) Write high byte to TABLAT
- 5) Execute TBLWT *+ (long write)

A long write to an internal EPROM location is terminated when an interrupt is received. The post increment TBLWT instruction is the only TBLWT instruction that is recommended for writes to int mal memory. (Writes to internal EPROM are only available on devices with 64 or more pins.)

TBLWT	Table Write	
Cycles:	2 (many if long write is to on-chip EPROM program memory)	

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation
No operation	No operation (Table Pointer on Address bus)	No operation	No operation (Table Latch on Address bus, WR goes low)

Example1:	TBLWT	*+;		•	•
Before Instructi TABLAT TBLPTR MEMORY(0	٠	= =	0x55 0x00A356 0xFF		
After Instruction TABLAT TBLPTR MEMORY(ns (table	=	completion) 0x55 0x00A357 0x55		
Example 2:	TBLWT	+*;			

Before Instruction

TABLAT = 0x34 TBLPTR = 0x01389A MEMORY(0x01389A) = 0xFF MEMORY(0x01389B) = 0xFF

After Instruction (table write completion)

TABLAT = 0x34
TBLPTR = 0x01389B
MEMORY(0x01389A) = 0xFF
MEMORY(0x01389B) = 0x34

Debugger Subroutine Cail RAP [label] TRAP iyntax: None)perands: $(PC) + 2 \rightarrow TOS$, Operation: 000028h → PC<20:1>

INBUG Status Affected:

Encoding:

0000 0000 1110 0000

Description:

Debugger Trap to 00028h. First, return address (PC+ 2) is pushed onto the return stack. Then the 20-bit value '000028h' is loaded into PC<20:1>. The INBUG status bit is set. TRAP is a two-cycle instruction.

Words:

1

Cycles:

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decod	Push PC to stack	No operation	Write 000028h to PC
No operation	No operation	No operation	No operation

Example:

HERE

TRAP

Before Instruction

Address (HERE) PC =

After Instruction

000028h

Address (HERE + 2)

PC = 0 TOS = A INBUG = 1

Trap Return from Subroutine TRET

Syntax:

[label] TRET

Operands:

Operation:

(TOS) → PC

PCLATU, PCLATH are unchanged

Status Affected:

INBUG

None

Encoding:

0000 0000 1110

Description:

Return from debugger trap. The stack is popped and the top of the stack (TOS) is loaded into the program counter. The

0001

INBUG status bit is cleared.

Words:

Cycles:

2

Q Cycle Activity:

Q1	Q2	Q3 ·	Q4
Decode	No operation	No operation	pop PC from stack
No operation	No operation	No operation	No operation

Example:

TRET

After Interrupt PC = TOS INBUG = 0

Test f, skip if 0 **TSTFSZ** [label] TSTFSZ f,a Syntax: $0 \le f \le 255$ Operands: $a \in [0,1]$ skip if f = 0Operation: None Status Affected: ffff ffff 0110 011a Encoding: If $\Upsilon = 0$, the next instruction, fetched Description: during the current instruction execution, is discarded and a NOP is executed making this a two-cycle instruction. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).

Words:

1

Cycles:

1(2)

Note: 3 cycles if skip and followed

by a 2-word instruction

Q Cycle Activity:

	Q3	<u> </u>
Read egister 'f'	Process Data	No operation
	Read egister 'f'	11000

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
operation			

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

HERE

TSTFSZ CNT, 1

NZERO

ZERO

Before Instruction

PC = Address(HERE)

After Instruction

If CNT

0x00,

Address (ZERO)

0x00. Address (NZERO)

Exclusiv OR literal with W **XORLW**

Syntax:

[label] XORLW k.

Operands:

 $0 \le k \le 255$

Operation:

(W) .XOR. $k \rightarrow W$

Status Affected:

N,Z

Encoding:

kkkk kkkk 1010 0000

Description:

The contents of W are XOR'ed with the 8-bit literal 'k'. The result is placed

in W.

Words:

1

Cycles:

1

Q Cycle Activity:

Q1	Q2 .	Q3	Q4
Decode	Read	Process	Write to W
	literal 'k'	Data	

Example:

XORLW 0xAF

Before Instruction

W

After Instruction

0x1A W

0xB5

Exclusiv OR W with f

CORWF Syntax:

[label] XORWF f,d,a

Operands:

 $0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$

Operation:

(W) .XOR. (f) \rightarrow dest

Status Affected:

N,Z

Encoding:

0001 10da ffff ffff

Description:

Exclusive OR the contents of W with register 'T. If 'd' is 0 the result is stored in W. If 'd' is 1 the result is stored back in the register 'T (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value

(default).

Words:

1

Cycles:

1

Q Cycle Activity:

Ω1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example:

XORWF REG, 1, 0

Before Instruction

 $\begin{array}{rcl}
REG & = & 0xAF \\
W & = & 0xB5
\end{array}$

After Instruction

REG = 0x1A W = 0xB5